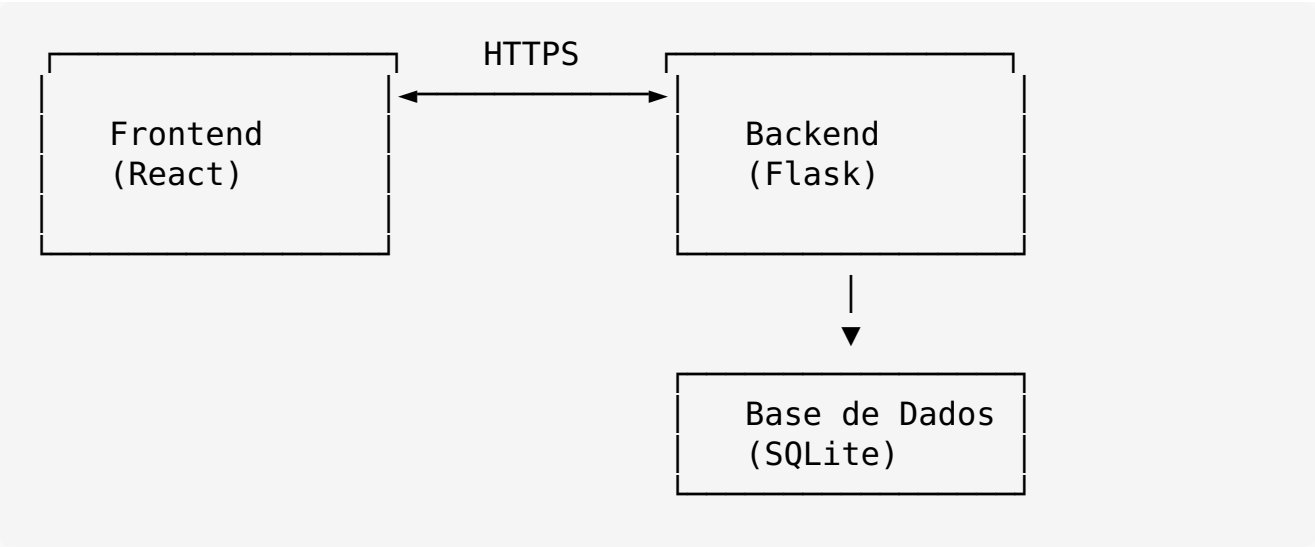
Guia Técnico - Sistema de Gestão Judicial

Índice

Arquitetura do Sistema

Visão Geral

O Sistema de Gestão Judicial segue uma arquitetura de aplicação web moderna com separação clara entre frontend e backend.



Componentes Principais

Frontend

- **Framework**: React 18 com Vite

- **UI Library**: Tailwind CSS + shadcn/ui
- **Routing**: React Router DOM
- **Icons**: Lucide React
- **Build Tool**: Vite

Backend

- **Framework**: Flask 3.0
- **ORM**: SQLAlchemy
- **Autenticação**: JWT (PyJWT)
- **CORS**: Flask-CORS
- **Base de Dados**: SQLite

Infraestrutura

- **Hosting**: Manus Cloud Platform
- **HTTPS**: Certificados automáticos
- **CDN**: Integrado na plataforma

Frontend - React

Estrutura do Projeto

```
tribunal_frontend/
├── public/
│   ├── index.html
│   └── assets/
├── src/
│   ├── components/
│   │   ├── ui/           # Componentes shadcn/ui
│   │   ├── Header.jsx    # Cabeçalho da aplicação
│   │   └── Footer.jsx    # Rodapé da aplicação
│   ├── pages/
│   │   ├── HomePage.jsx      # Página inicial
│   │   ├── CaseSearchPage.jsx # Pesquisa de processos
│   │   ├── FormsPage.jsx     # Formulários
│   │   └── LoginPage.jsx     # Autenticação
│   ├── App.jsx          # Componente principal
│   ├── App.css          # Estilos globais
│   └── main.jsx         # Ponto de entrada
├── package.json
└── vite.config.js
```

Componentes Principais

App.jsx

```
 import { BrowserRouter as Router, Routes, Route } from 'react-
router-dom'
import Header from './components/Header'
import Footer from './components/Footer'
import HomePage from './pages/HomePage'
import CaseSearchPage from './pages/CaseSearchPage'
import FormsPage from './pages/FormsPage'
import LoginPage from './pages/LoginPage'

function App() {
  return (
    <Router>
      <div className="min-h-screen flex flex-col">
        <Header />
        <main className="flex-1">
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/cases" element={<CaseSearchPage />} />
            <Route path="/forms" element={<FormsPage />} />
            <Route path="/login" element={<LoginPage />} />
          </Routes>
        </main>
        <Footer />
      </div>
    </Router>
  )
}
```

Gestão de Estado

- **Local State**: useState para componentes individuais
- **Authentication**: localStorage para tokens JWT
- **API Calls**: fetch nativo com async/await

Styling

- **Tailwind CSS**: Framework de utilidades
- **shadcn/ui**: Componentes pré-construídos
- **Responsive Design**: Mobile-first approach

APIs Frontend

Configuração Base

```
const API_BASE_URL = 'https://kkh7ikcgn7ol.manus.space'

const apiCall = async (endpoint, options = {}) => {
  const response = await fetch(`${API_BASE_URL}${endpoint}`, {
    headers: {
      'Content-Type': 'application/json',
      ...options.headers
    },
    ...options
  })
  return response.json()
}
```

Autenticação

```
const login = async (credentials) => {
  const response = await apiCall('/api/auth/login', {
    method: 'POST',
    body: JSON.stringify(credentials)
  })

  if (response.success) {
    localStorage.setItem('token', response.token)
    localStorage.setItem('user', JSON.stringify(response.user))
  }

  return response
}
```

Backend - Flask

Estrutura do Projeto

```
tribunal_backend/
├── src/
│   ├── models/
│   │   └── user.py       # Modelos de dados
│   ├── routes/
│   │   ├── auth.py       # Autenticação
│   │   ├── case.py       # Processos
│   │   └── form.py       # Formulários
```

```
│   └── main.py           # Aplicação principal
├── requirements.txt
├── create_sample_data.py # Script de dados de exemplo
└── venv/                 # Ambiente virtual
```

Aplicação Principal (main.py)

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
import os

# Configuração da aplicação
app = Flask(__name__)
app.config['SECRET_KEY'] = 'tribunal-secret-key-2024'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///tribunal.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Inicialização de extensões
db = SQLAlchemy(app)
CORS(app, origins="*")

# Importação de rotas
from routes.auth import auth_bp
from routes.case import case_bp
from routes.form import form_bp

# Registo de blueprints
app.register_blueprint(auth_bp, url_prefix='/api/auth')
app.register_blueprint(case_bp, url_prefix='/api/cases')
app.register_blueprint(form_bp, url_prefix='/api/forms')

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(host='0.0.0.0', port=5000, debug=False)
```

Modelos de Dados

User Model

```python
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
import hashlib

db = SQLAlchemy()

class User(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True,
nullable=False)
    password_hash = db.Column(db.String(120), nullable=False)
    role = db.Column(db.String(20), nullable=False,
default='user')
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def set_password(self, password):
        self.password_hash =
hashlib.sha256(password.encode()).hexdigest()

    def check_password(self, password):
        return self.password_hash ==
hashlib.sha256(password.encode()).hexdigest()
```

Case Model

```
class Case(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    case_number = db.Column(db.String(20), unique=True,
nullable=False)
    title = db.Column(db.String(200), nullable=False)
    case_type = db.Column(db.String(20), nullable=False)
    status = db.Column(db.String(20), nullable=False)
    plaintiff = db.Column(db.String(100), nullable=False)
    defendant = db.Column(db.String(100), nullable=False)
    judge = db.Column(db.String(100))
    filing_date = db.Column(db.Date, nullable=False)
    next_hearing = db.Column(db.Date)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
```

Rotas e Endpoints

Autenticação (auth.py)

```
from flask import Blueprint, request, jsonify
import jwt
from datetime import datetime, timedelta

auth_bp = Blueprint('auth', __name__)

@auth_bp.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')
```

```
    user = User.query.filter_by(username=username).first()

    if user and user.check_password(password):
        token = jwt.encode({
            'user_id': user.id,
            'username': user.username,
            'role': user.role,
            'exp': datetime.utcnow() + timedelta(hours=24)
        }, app.config['SECRET_KEY'], algorithm='HS256')

        return jsonify({
            'success': True,
            'token': token,
            'user': {
                'id': user.id,
                'username': user.username,
                'role': user.role
            }
        })

    return jsonify({'success': False, 'error': 'Credenciais
inválidas'}), 401
```

Pesquisa de Processos (case.py)

```
@case_bp.route('/search', methods=['GET'])
def search_cases():
    # Parâmetros de pesquisa
    case_number = request.args.get('case_number', '')
    party_name = request.args.get('party_name', '')
    case_type = request.args.get('case_type', '')
    status = request.args.get('status', '')

    # Paginação
    page = int(request.args.get('page', 1))
    per_page = int(request.args.get('per_page', 10))

    # Query base
    query = Case.query

    # Aplicar filtros
    if case_number:
        query =
query.filter(Case.case_number.contains(case_number))
    if party_name:
        query = query.filter(
            db.or_(
                Case.plaintiff.contains(party_name),
                Case.defendant.contains(party_name)
            )
```

```
        )
    if case_type:
        query = query.filter(Case.case_type == case_type)
    if status:
        query = query.filter(Case.status == status)

    # Executar query com paginação
    pagination = query.paginate(
        page=page, per_page=per_page, error_out=False
    )

    return jsonify({
        'success': True,
        'cases': [case.to_dict() for case in pagination.items],
        'pagination': {
            'page': page,
            'per_page': per_page,
            'total': pagination.total,
            'pages': pagination.pages,
            'has_next': pagination.has_next,
            'has_prev': pagination.has_prev
        }
    })
```

Base de Dados

Esquema da Base de Dados

Tabela: users

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username VARCHAR(80) UNIQUE NOT NULL,
    password_hash VARCHAR(120) NOT NULL,
    role VARCHAR(20) NOT NULL DEFAULT 'user',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Tabela: cases

```
CREATE TABLE cases (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    case_number VARCHAR(20) UNIQUE NOT NULL,
    title VARCHAR(200) NOT NULL,
    case_type VARCHAR(20) NOT NULL,
    status VARCHAR(20) NOT NULL,
```

```
    plaintiff VARCHAR(100) NOT NULL,
    defendant VARCHAR(100) NOT NULL,
    judge VARCHAR(100),
    filing_date DATE NOT NULL,
    next_hearing DATE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Tabela: forms

```
CREATE TABLE forms (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    category VARCHAR(50) NOT NULL,
    version VARCHAR(10) NOT NULL,
    file_path VARCHAR(255),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Dados de Exemplo

Utilizadores

```
users = [
    {'username': 'admin', 'password': 'admin123', 'role':
'admin'},
    {'username': 'juiz_silva', 'password': 'juiz123', 'role':
'judge'}
]
```

Processos

```
cases = [
    {
        'case_number': '2024/CV/001',
        'title': 'Silva vs. Santos - Ação de Cobrança',
        'case_type': 'civil',
        'status': 'open',
        'plaintiff': 'João Silva',
        'defendant': 'Maria Santos',
        'judge': 'juiz_silva',
        'filing_date': '2025-05-19',
        'next_hearing': '2025-07-03'
    }
]
```

APIs e Endpoints

Documentação da API

Base URL

```
https://kkh7ikcgn7ol.manus.space/api
```

Autenticação

POST /auth/login

Autentica um utilizador e retorna um token JWT.

Request:

```
{
    "username": "admin",
    "password": "admin123"
}
```

Response:

```
{
    "success": true,
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
    "user": {
        "id": 1,
        "username": "admin",
        "role": "admin"
    }
}
```

Processos

GET /cases/search

Pesquisa processos com filtros opcionais.

Parâmetros: - `case_number` (string): Número do processo - `party_name` (string): Nome da parte - `case_type` (string): Tipo de processo - `status` (string): Estado do processo - `page` (int): Página (default: 1) - `per_page` (int): Itens por página (default: 10)

Response:


```
{
    "success": true,
    "cases": [
        {
            "id": 1,
            "case_number": "2024/CV/001",
            "title": "Silva vs. Santos - Ação de Cobrança",
            "case_type": "civil",
            "status": "open",
            "plaintiff": "João Silva",
            "defendant": "Maria Santos",
            "judge": "juiz_silva",
            "filing_date": "2025-05-19",
            "next_hearing": "2025-07-03"
        }
    ],
    "pagination": {
        "page": 1,
        "per_page": 10,
        "total": 3,
        "pages": 1,
        "has_next": false,
        "has_prev": false
    }
}
```


GET /cases/types

Retorna os tipos de processo disponíveis.

Response:


```
{
    "success": true,
    "case_types": [
        {"value": "civil", "label": "Civil"},
        {"value": "criminal", "label": "Criminal"},
        {"value": "family", "label": "Família"},
        {"value": "probate", "label": "Sucessões"}
    ]
}
```


Formulários

GET /forms

Lista formulários com filtros opcionais.

Parâmetros: - `category` (string): Categoria do formulário - `search` (string): Termo de pesquisa

Response:

```
{
    "success": true,
    "forms": {
        "civil": [
            {
                "id": 1,
                "title": "Petição Inicial - Ação Declarativa",
                "description":
"Formulário para iniciar uma ação declarativa cível",
                "version": "2.1"
            }
        ]
    }
}
```

GET /forms/categories

Retorna as categorias de formulários.

Response:

```
{
    "success": true,
    "categories": [
        {"value": "civil", "label": "Formulários Cíveis"},
        {"value": "criminal", "label": "Formulários Criminais"},
        {"value": "family", "label": "Formulários de Família"},
        {"value": "probate", "label": "Formulários de
Sucessões"}
    ]
}
```

Deployment e Infraestrutura

Ambiente de Produção

URLs

- **Frontend**: https://glmwuefy.manus.space
- **Backend**: https://kkh7ikcgn7ol.manus.space

Plataforma

- **Hosting**: Manus Cloud Platform
- **SSL**: Certificados automáticos
- **CDN**: Distribuição global
- **Backup**: Automático

Processo de Deploy

Frontend

```
# Build da aplicação
cd tribunal_frontend
pnpm run build

# Deploy automático
manus-deploy-frontend react ./
```

Backend

```
# Preparação do ambiente
cd tribunal_backend
source venv/bin/activate
pip install -r requirements.txt

# Deploy automático
manus-deploy-backend flask ./
```

Configurações de Produção

Frontend (vite.config.js)

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import path from 'path'

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
    },
  },
  build: {
    outDir: 'dist',
```

```
    sourcemap: false,
    minify: 'terser'
  }
})
```

Backend (main.py)

```
# Configurações de produção
app.config['DEBUG'] = False
app.config['TESTING'] = False
app.config['SECRET_KEY'] = os.environ.get('SECRET_KEY',
'fallback-key')

# CORS para produção
CORS(app, origins=[
    "https://glmwuefy.manus.space",
    "https://*.manus.space"
])

# Servidor de produção
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=False)
```

Segurança

Implementações Atuais

HTTPS

- Certificados SSL automáticos
- Redirecionamento HTTP → HTTPS
- Headers de segurança

Autenticação

- JWT tokens com expiração (24h)
- Hashing de passwords (SHA-256)
- Validação de credenciais

CORS

- Configurado para domínios específicos
- Headers permitidos controlados
- Métodos HTTP restringidos

Validação de Dados

- Sanitização de inputs
- Validação de parâmetros
- Tratamento de erros

Recomendações Futuras

Segurança Avançada

1. **Rate Limiting** ```python from flask_limiter import Limiter

limiter = Limiter(app, key_func=get_remote_address, default_limits=["200 per day", "50 per hour"]) ```

1. **Logs de Auditoria** ```python import logging

logging.basicConfig(filename='audit.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s') ```

1. **Validação Robusta** ```python from marshmallow import Schema, fields, validate

class LoginSchema(Schema): username = fields.Str(required=True, validate=validate.Length(min=3)) password = fields.Str(required=True, validate=validate.Length(min=6)) ```

Manutenção e Monitorização

Logs e Monitorização

Frontend

- Console logs para debugging
- Error boundaries para captura de erros
- Performance monitoring via browser tools

Backend

- Flask logging integrado
- Error handling personalizado
- Request/response logging

Plataforma

- Monitorização automática Manus
- Alertas de disponibilidade
- Métricas de performance

Backup e Recuperação

Base de Dados

```
# Backup manual
sqlite3 tribunal.db ".backup backup_$(date +%Y%m%d).db"

# Restauro
sqlite3 tribunal.db ".restore backup_20241218.db"
```

Código

- Versionamento no ambiente de desenvolvimento
- Backup automático da plataforma
- Rollback disponível

Atualizações

Frontend

1. Fazer alterações no código
2. Testar localmente
3. Build da aplicação
4. Deploy automático

Backend

1. Fazer alterações no código
2. Testar APIs localmente
3. Atualizar requirements.txt se necessário
4. Deploy automático

Desenvolvimento Futuro

Funcionalidades Planeadas

Curto Prazo (1-3 meses)

1. **Calendário de Audiências**
2. Interface de consulta
3. Filtros por juiz e data
4. Integração com sistema de processos
5. **Notificações**
6. Sistema de alertas
7. Email notifications
8. Push notifications
9. **Dashboard Administrativo**
10. Estatísticas de utilização
11. Gestão de utilizadores
12. Relatórios de sistema

Médio Prazo (3-6 meses)

1. **API Móvel**
2. Endpoints otimizados
3. Autenticação móvel
4. Sincronização offline
5. **Integração Externa**
6. APIs de terceiros
7. Sistemas legados
8. Importação de dados
9. **Relatórios Avançados**
10. Geração de PDFs
11. Gráficos e estatísticas
12. Exportação de dados

Longo Prazo (6+ meses)

1. **Inteligência Artificial**
2. Pesquisa semântica
3. Classificação automática
4. Assistente virtual
5. **Microserviços**
6. Separação de responsabilidades
7. Escalabilidade horizontal
8. Containerização

Melhorias Técnicas

Performance

- Caching de dados
- Lazy loading
- Otimização de queries
- CDN para assets

Escalabilidade

- Load balancing
- Database clustering
- Horizontal scaling
- Microservices architecture

Monitorização

- APM (Application Performance Monitoring)
- Real-time alerts
- Custom dashboards
- Log aggregation

Este guia técnico fornece uma visão abrangente da arquitetura e implementação do Sistema de Gestão Judicial. Para questões específicas de desenvolvimento, consulte a documentação do código ou contacte a equipa técnica.